Gradient of Multi-class Cross-Entropy

The first step on the backpropagation is to have the gradient of the loss w.r.t to the model/hypothesis output (ie: Scores on the neural network output layer). On this text I try to explain how to get this gradient for the multi class cross-entropy, which can be consider as the "default" loss for multi-class problems.

Sigmoid activation function

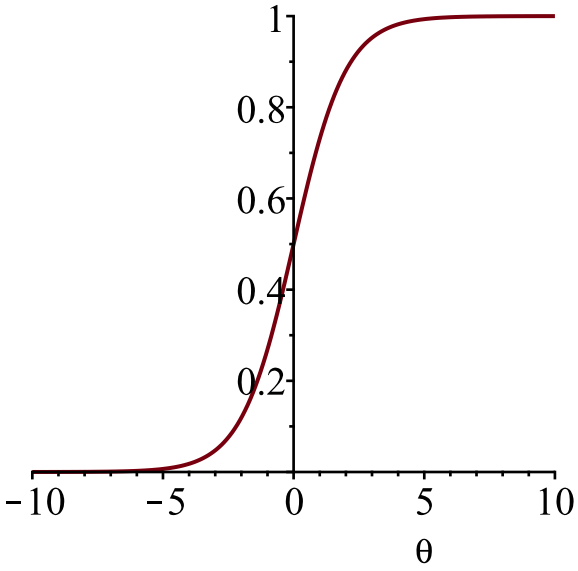
Before we start let's review the sigmoid activation function that output

$S(t) = \frac{1}{1 + e^{-\theta}}$, where θ will be for instance the result of a previous layer, for example $\theta = w^T.x + b$

$$sigmoid_{activation} := (\theta) \rightarrow \frac{1}{1 + \exp(-\theta)}$$

$$\theta \rightarrow \frac{1}{1 + e^{-\theta}} \quad \textbf{(1)}$$

2D Plot 1/(1C exp(-theta))



If we want to calculate the derivative of the sigmoid w.r.t to θ

$$\frac{\partial}{\partial \theta}\left(\frac{1}{1 + \exp(-\theta)}\right) = \frac{e^{-\theta}}{(1 + e^{-\theta})^2}$$

to matlab

```
cg = 0.1e1 / (0.1e1 + exp(-theta)) ^ 2 * exp(-theta);
```

Softmax activation function

It's an activation function that converts the scores vector from the last linear layer (FC), and convert to probabilities. It's considered to be a generalization of the logistic function (Sigmoid).

$$S(a) := \begin{bmatrix} a_1 \\ a_2 \\ a_n \end{bmatrix} \rightarrow \begin{bmatrix} p_1 \\ p_2 \\ p_n \end{bmatrix}$$

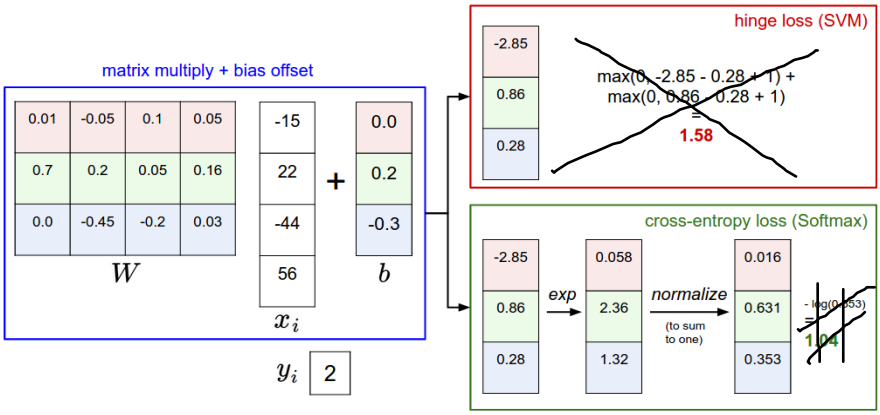
$$\sigma_z = \frac{\exp(z_j)}{\sum_{k=1}^{N} \exp(z_k)}, \textbf{ for } j = [1, 2, 3, ...etc], \text{ where j is an index to each possible class N}$$

In other words the softmax activation function will convert the scores to probabilities by dividing the exponential of each value on the score (z) vector by the sum of all those exponentials.
The idea is that the softmax will transform the scores into probabilities.
During the forward propagation we will calculate the softmax activation for each value of our score vector, converting them to probabilities.
During training (backpropagation) we will be more interested on the probability of the correct class p_k

Now let's define the softmax function (note that exp~, means exponential for each element). Here z is the score vector and N the vector size.

$$softmax_{activation} := (z, N) \rightarrow \frac{\exp\sim(z)}{sum(\ \exp\sim(\text{'}z[i]\text{'}), i = 1\ ..N\ )} = (z, N) \rightarrow \frac{\text{`}\sim\text{`}\exp(z)}{\sum_{i=1}^{N} \text{`}\sim\text{`}\exp(\text{'}z_i\text{'})}$$

Let's implement the following example, considering the softmax path, and disconsidering the cross-entropy (log) calculation:



$scores := Vector([-2.85, 0.86, 0.28]) = \begin{bmatrix} -2.85 \\ 0.86 \\ 0.28 \end{bmatrix}$, then calculating the softmax on the whole vector

$softmax_{activation}(scores, \text{numelems}(\text{scores})) = \begin{bmatrix} 0.0154493156751271 \\ 0.631163353697413 \\ 0.353387330637003 \end{bmatrix}$

On simpler models where we use binary classification we can use the cross-entropy function from the output of our logistic regression activation.(ie: Logistic regresion model).

$$H(p, q) = -\sum_i p_i \log(q_i)$$

Here p_i are the labels and q_i the model/hypothesis probabilities, i is the index for each possible element of our labels and probabilities.

The model/hypothesis probabilities are given by using the logistic function:

$q_i = g(z) = \frac{1}{1 + \exp(-z)}$, where input z will be the output of your layer. z=w.x

Now the outpuut of your network will be complementary probabilities so Σ(qi)=1

Probability of output y=1, here q is the output of the model

$q_{y=1} \equiv \hat{y} \equiv g(w.x)$

Probability of output y=0

$q_{y=0} \equiv 1 - \hat{y}$

Now considering p the true probability (true label)
$p \in \{y, 1-y\}$ **and** $q \in \{\hat{y}, 1-\hat{y}\}$

Plugin now p and q on the cross-entropy function we have:

$$H(p,q) = -\sum_i p_i.\log(q_i) \equiv -(y.\log(\hat{y}) + (1-y).\log(1-\hat{y})) \equiv -y.\log(\hat{y}) - (1-y).\log(1-\hat{y})$$

Now on a training set of N samples we define the cross-entropy loss function as

$$L(W) = \frac{1}{N}.\sum_{n=1}^{N} H(p_n, q_n) \equiv -\frac{1}{N}.\sum_{n=1}^{N}\left[y.\log(\hat{y}_n) + (1-y_n).\log(1-\hat{y}_n)\right]$$

where y_n is the n-th label and $\hat{y}_n$ is the n-th score.

Multi-class case

On the case for multi-class you can use a different version of the cross-entropy function called multi-class cross-entropy or negative log likelyhood. This function is just a generalization of the cross-entropy.

$$H(p,q) = -\sum_i p_i.\log(q_i)$$

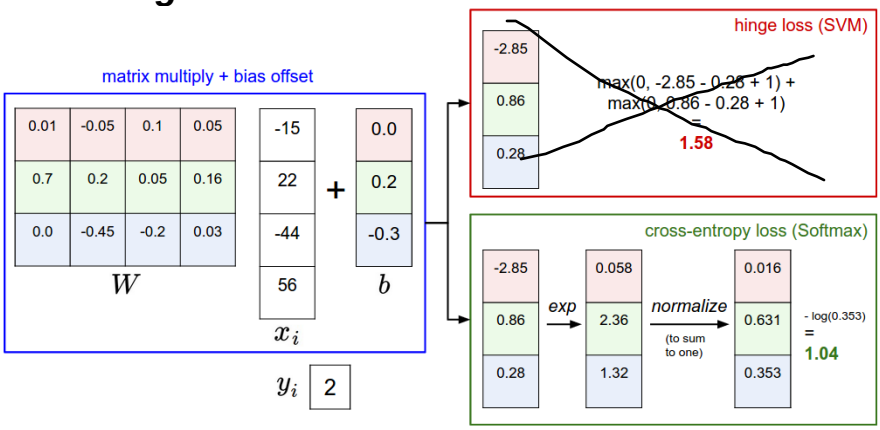
Now p(true probabilities from labels) will be vector (one-hot encoded), and q will be the output of the softmax function, which are already normalized probabilities.
Due to the fact that we're using the one-hot encoding only one element will be one all the others will be zero, so we can simplify:

$L_i = -\log(q_i)$, here q_i is the output of the softmax function at the position of the correct label

$$q_z = \frac{\exp(z_j)}{\sum_{k=1}^{N}\exp(z_k)}, \textbf{ for } j = [1, 2, 3, ...etc]$$

$$L_i = -\log\left(\frac{\exp\left(z_{y_i}\right)}{\sum_{k=1}^{N}\exp(z_k)}\right)$$, again z_{y_i}, is the score referent to the correct class

Calculating loss:



$$L_i := (z, N, y_i) \to -\log\left(\frac{\exp\sim(z[y_i])}{sum(\ \exp\sim('z[i]'), i = 1\ ..N\ )}\right) = (z, N, y_i) \to -\log\left(\frac{\text{`}\sim\text{`}\exp\left(z_{y_i}\right)}{\sum_{i=1}^{N}\text{`}\sim\text{`}\exp('z_i')}\right)$$

$$scores := Vector([-2.85, 0.86, 0.28]) = \begin{bmatrix} -2.85 \\ 0.86 \\ 0.28 \end{bmatrix}$$

Now considering the index of the correct class at 3

$L_i(scores, \text{numelems}(\text{scores}), 3) = 1.040190570$

Gradient of the multi class cross-entropy loss

The first step of the backpropagation is to calculate the gradient of the loss function w.r.t to it's inputs. On this case the input's of the loss function is the score of the correct class.
Putting again our loss

$$L_i = -\log\left(\frac{\exp(z_i)}{\sum_{k=1}^{N}\exp(z_k)}\right)$$ and what we want is: $\frac{\partial}{\partial z_i} L_i$

As our loss is already a composition of 2 functions (cross-entropy and softmax) we can easily use the chain-rule:

$L_i = -\log(q_i)$ **and** $q_i = \frac{\exp(z_i)}{\sum_{k=1}^{N}\exp(z_k)}$, where q_i is the probability of the correct class and z_i the score of the correct class

$$\frac{\partial}{\partial z_i} L_i = \frac{\partial}{\partial q_i} L_i \cdot \frac{\partial}{\partial z_i} q_i$$

Now the first derivative $\frac{\partial}{\partial q_i} L_i$

$$-\frac{\mathrm{d}}{\mathrm{d}\, q_i}\log(q_i) = -\frac{1}{q_i}$$

The second derivative, $\frac{\partial}{\partial z_i}(q_i)$ where $q_i = \frac{\exp(z_i)}{\sum_{k=1}^{N}\exp(z_k)}$, will be a little more complicated, before continue let's remember some stuff:

$$\frac{\mathrm{d}}{\mathrm{d}\, x}\left(\frac{f(x)}{g(x)}\right) = \frac{f'(x)\cdot g(x) - g'(x)\cdot f(x)}{(g(x))^2}$$

$$\frac{\mathrm{d}}{\mathrm{d}\, x}\left(e^x\right) = e^x$$

$$\frac{\mathrm{d}}{\mathrm{d}\, x}(\log(x)) = \frac{1}{x}$$

$$\frac{\mathrm{d}}{\mathrm{d}\, x}\left(\sum_k a_k x^3\right) = \sum_k\left(3.\, a_k x^2\right)$$

Also to make simpler let's consider $\sum_{k=1}^{N} \exp(z_k) = \Sigma_c(x)$, the derivative will become:

$$\frac{d}{d z_i}\left(\frac{\exp(z_i)}{\Sigma_c(z_i)}\right)$$

$$\frac{e^{z_i}}{\Sigma_c(z_i)} - \frac{e^{z_i}\left(\frac{d}{dz_i}\Sigma_c(z_i)\right)}{\Sigma_c(z_i)^2} \quad \textbf{(2)}$$

Now paying more attention to the term $\frac{d}{dz_i}\Sigma_c(z_i)$ and expanding we have $\frac{d}{d z_i}\left(\sum_{k=1}^{N} \exp(z_k)\right)$, which means that we want to know the derivative of this sum when a particular element z_i changes, for example considering N=3, and i=2

$$\frac{d}{d z_i}\left(\exp(a) + \exp(z_i) + \exp(c)\right)$$

$$e^{z_i} \quad \textbf{(3)}$$

Now we will substitute **(3)** on **(2)** to substitute $\frac{d}{dz_i}\Sigma_c(z_i)$ by e^{z_i} on equation **(2)**

$$subs\left(\frac{d}{dz_i}\Sigma_c(z_i) = \textbf{(3)}, \textbf{(2)}\right)$$

$$\frac{e^{z_i}}{\Sigma_c(z_i)} - \frac{\left(e^{z_i}\right)^2}{\Sigma_c(z_i)^2} \quad \textbf{(4)}$$

$$subs\left(\Sigma_c(z_i) = \Sigma_c, \textbf{(4)}\right)$$

$$\frac{e^{z_i}}{\Sigma_c} - \frac{\left(e^{z_i}\right)^2}{\Sigma_c^2} \quad \textbf{(5)}$$

$$subs\left(\frac{e^{z_i}}{\Sigma_c} = p_k, \textbf{(5)}\right)$$

$$p_k - \frac{\left(e^{z_i}\right)^2}{\Sigma_c^2} \quad \textbf{(6)}$$

$$subs\left(\frac{\left(e^{z_i}\right)^2}{\Sigma_c^2} = \left(p_k^2\right), \textbf{(6)}\right) = -p_k^2 + p_k \overset{\text{factor}}{=} -p_k\left(p_k - 1\right)$$

here p_k is the probability of the correct class.